理解 FTRL 算法

05 December 2015

引言

在工业界，越来越多的业务需要大规模机器学习，不单参与训练的数据量大，模型特征量的规模也大。例如点击率预估，训练数据量在TB量级，特征量在亿这个量级，业内常用LR（Logistic Regression）和FM（Factorization Machines）为点击率预估建模。对LR、FM这类模型的参数学习，传统的学习算法是batch learning算法，它无法有效地处理大规模的数据集，也无法有效地处理大规模的在线数据流。这时，有效且高效的online learning算法显得尤为重要。

SGD算法[1]是常用的online learning算法，它能学习出不错的模型，但学出的模型不是稀疏的。为此，学术界和工业界都在研究这样一种online learning算法，它能学习出有效的且稀疏的模型。FTRL（Follow the Regularized Leader）算法正是这样一种算法，它由Google的H. Brendan McMahan在2010年提出的[2]，作者后来在2011年发表了一篇关于FTRL和AOGD、FOBOS、RDA比较的论文[3]，2013年又和Gary Holt, D. Sculley, Michael Young等人发表了一篇关于FTRL工程化实现的论文[4]。如论文[4]的内容所述，FTRL算法融合了RDA算法能产生稀疏模型的特性和SGD算法能产生更有效模型的特性。它在处理诸如LR之类的带非光滑正则化项（例如1范数，做模型复杂度控制和稀疏化）的凸优化问题上性能非常出色，国内各大互联网公司都已将该算法应用到实际产品中。

文章[5]是篇很好的介绍FTRL算法的中文资料，从TG算法、FOBOS算法开始，到RDA算法，最后到FTRL算法，一脉相承，而且各个算法都有推导过程，值得认真体会。

这篇文章不会赘述文章[5]中的内容，而是介绍FTRL算法与SGD算法之间存在的另一种联系。这个联系在网络上似乎没有文章介绍，可能是因为这个细节不那么重要，但倘若了解了此细节，能更好的体会到FTRL算法为啥跟SGD算法有联系。

FTRL算法与SGD算法的联系

SGD算法的迭代计算公式如下：

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \mathbf{g}_t \tag{1}$$

其中 *t* 为迭代轮数，w是模型参数，g是loss function关于w的梯度，而η是学习率，它随着迭代轮数增多而递减。

FTRL算法的迭代算公式如下：

$$\mathbf{w}_{t+1} = \arg\min_{\mathbf{w}} \left(\sum_{s=1}^{t} \mathbf{g}_s \cdot \mathbf{w} + \frac{1}{2} \sum_{s=1}^{t} \sigma_s ||\mathbf{w} - \mathbf{w}_s||_2^2 + \lambda_1 ||\mathbf{w}||_1 \right) \tag{2}$$

其中 *t* 为迭代轮数，w是模型参数，σ_s 定义成 $\sum_{s=1}^{t} \sigma_s = \frac{1}{\eta_t}$ λ_1是L1正则化系数。

公式2看似比公式1复杂，需要求最优解，但其实很容易算出 $\mathbf{w}_{t+1}$ 的close form表达式，详见文章[4]。在公式2中，$argmin$ 算子的内容中由3项组成，最后一项是L1正则（当然也可以再加上L2正则），很明显L1正则是为了获取稀疏模型。如果令 $\lambda_1 = 0$，也就是说不要正则项，公式2就变成下面的公式3：

$$\mathbf{w}_{t+1} = \arg\min_{\mathbf{w}} \left(\sum_{s=1}^{t} \mathbf{g}_s \cdot \mathbf{w} + \frac{1}{2} \sum_{s=1}^{t} \sigma_s ||\mathbf{w} - \mathbf{w}_s||_2^2 \right) \tag{3}$$

大家都能体会到这是FTRL算法的核心，但更重要的是，公式3就直接等价于公式1。强调一下，是等价，而不是近似。这点正是本节要描述的联系，下文对这点进行证明。

首先，记公式3右侧arg min算子的内容为f(w)，显然f(w)是凸函数，所以公式3存在极值。将其对w求梯度，得到

$$\frac{\partial f(\mathbf{w})}{\partial \mathbf{w}} = \sum_{s=1}^{t} \mathbf{g}_s + \sum_{s=1}^{t} \sigma_s (\mathbf{w} - \mathbf{w}_s) \tag{4}$$

梯度是个向量，令公式4等于零向量，这时即可得极值，极值正是 $\mathbf{w}_{t+1}$。这样也就是有下面的公式5成立：

$$\sum_{s=1}^{t} \mathbf{g}_s + \sum_{s=1}^{t} \sigma_s (\mathbf{w}_{t+1} - \mathbf{w}_s) = 0 \tag{5}$$

将含有wt+1的项放到等号左边，剩下的放在右边，得到公式6：

$$(\sum_{s=1}^{t} \sigma_s) \mathbf{w}_{t+1} = \sum_{s=1}^{t} \sigma_s \mathbf{w}_s - \sum_{s=1}^{t} \mathbf{g}_s \tag{6}$$

进一步化简得到公式7：

$$\frac{1}{\eta_t} \mathbf{w}_{t+1} = \sum_{s=1}^{t} \sigma_s \mathbf{w}_s - \sum_{s=1}^{t} \mathbf{g}_s \tag{7}$$

用 *t-1* 替换 *t*，得到公式8：

$$\frac{1}{\eta_{t-1}} \mathbf{w}_t = \sum_{s=1}^{t-1} \sigma_s \mathbf{w}_s - \sum_{s=1}^{t-1} \mathbf{g}_s \tag{8}$$

用公式7减去公式8，即式子的左边右边同时减去，得到公式9：

$$\frac{1}{\eta_t} \mathbf{w}_{t+1} - \frac{1}{\eta_{t-1}} \mathbf{w}_t = \sigma_t \mathbf{w}_t - \mathbf{g}_t \tag{9}$$

把 σ_t 用 η_t 表示，得到公式10：

$$\frac{1}{\eta_t} \mathbf{w}_{t+1} - \frac{1}{\eta_{t-1}} \mathbf{w}_t = (\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}}) \mathbf{w}_t - \mathbf{g}_t \tag{10}$$

对公式10化简即可得到公式1。

通过上面的推导证明，我们看到公式3与公式1确实等价。由此可以更好的体会下公式2为啥长这样子，其左边两项承担了SGD算法的功能，而最右边的一项承担的是得到稀疏模型的功能。因为有这样的一种数学含义在背后，所以才好放心的下结论说，FTRL算法融合了RDA算法能产生稀疏模型的特性和SGD算法能产生更有效模型的特性，也就是说能学习出有效的且稀疏的模型。

参考文献

[1] Stochastic gradient descent (From Wikipedia)

[2] H. Brendan McMahan & M Streter. Adaptive Bound Optimization for Online Convex Optimization. In COLT, 2010

[3] H. Brendan McMahan. Follow-the-Regularized-Leader and Mirror Descent: Equivalence Theorems and L1 Regularization. In AISTATS, 2011

[4] H. Brendan McMahan, Gary Holt, D. Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, Sharat Chikkerur, Dan Liu, Martin Wattenberg, Arnar Mar Hrafnkelsson, Tom Boulos, Jeremy Kubica, Ad Click Prediction: a View from the Trenches. In ACM SIGKDD, 2013

[5] 冯扬, 在线最优化求解

注：原文地址：理解 FTRL 算法